Temporary headers in VolksForth

Philip Zembrod (pzembrod@gmail.com)
SVFIG Meeting April 24th, 2021

Introduction

Project: Write a Small C compiler on the C64 (late 80s)

- 1st approach with macro asm lib: concern about code size
- search for better language led to VolksForth

Grew to love it. Took design lessons from Thinking Forth

- components with terminology

Code size remained a concern

- wondered how Hejlsberg achieved Turbo Pascal ...

VolksForth: Headerless words in turnkey application

```
| : hello ." Hello" cr ;
```

| places following header “on the heap”

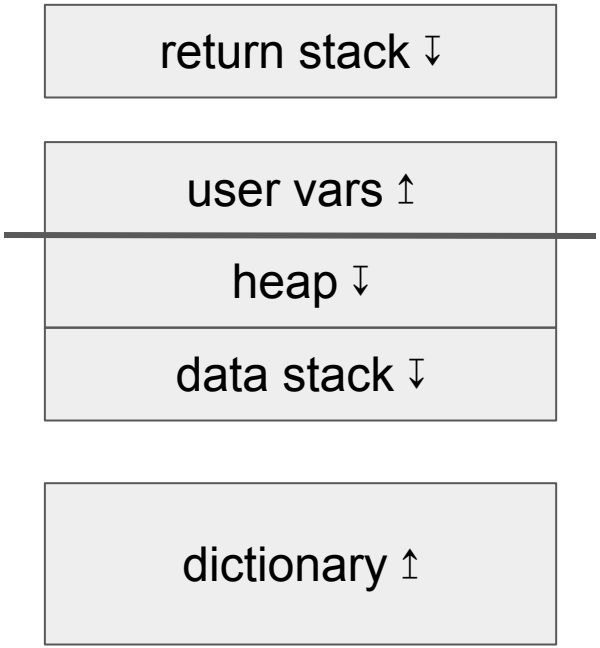
“Heap” is no C-style heap

… more like a downward growing temp dictionary

Terminology:

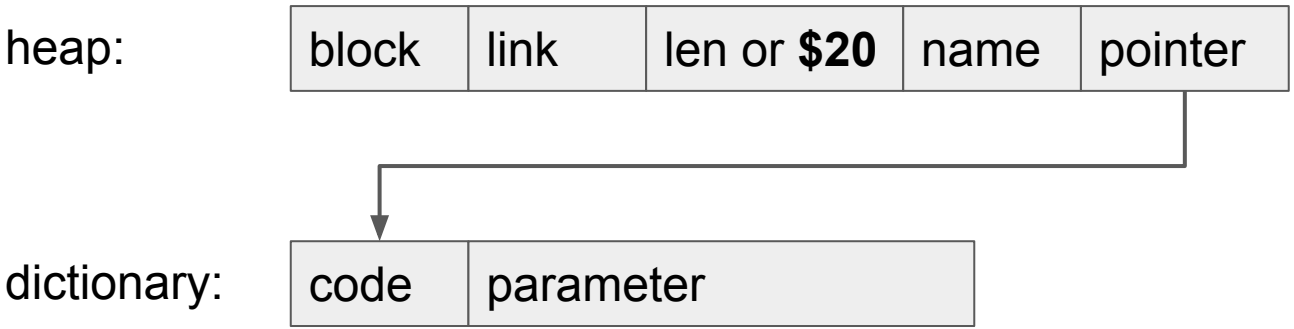
```
heap ( -- addr )   hallot ( n -- )   clear ( -- )   | ( -- )
```

Memory map




```
: hallot ( quan -- )
 s0 @ over - swap
 sp@ 2+  dup rot -  dup s0 !
 2 pick over -  move  clearstack
 s0 ! ;
```

Word header with “indirect” bit $20



Also used for ALIAS:

```
‘ 1+ alias plus-one
```

restrict:	$80
immediate:	$40
indirect:	$20

Transient assembler

```
here   $800 hallot  heap dp !
include 6502asm.fth
dp !
```

- Turnkey application
  - with code words
  - without assembler

Memory gain for cc64

WORDS shows `|prefix`

```
CC  |SCRATCH-EXE  |,!  |S!  |WAIT-SCRATCH  |?USAGE  |PASS1  |PASS2  |LINK-EXE
|LINK-LIB  |WRITE-DECLARATIONS  |WRITE-LIBHEADER  |LINK-LIBSTATICS
|(LINK-LIB  |LINK-STATICS  |(LINK-STATICS  |LINK-CODE  |P2PC  |LINK-RUNTIMEMODULE
… … ...
|*DOUBLEDEF*  |*SYMOVFL*  |*EOF*  |*SYNTAX*  |ERRORMESSAGE
|FINDSTR  |TOKEN  |STRING[]  |+STRING  |>STRING
|ENDTAB  |STRINGTAB  |VOID  |NIL  |X"  |X  |M  |N  |INIT:  |INIT  |INITS
```

- `clear` frees ~14 kB (~800 names + 6502asm)
- comparison: ~18 kB code, ~14 kB Forth core

Commander X16: tighter memory

Commander X16: new 65C02 retro computer

www.commanderx16.com

Contiguous RAM:

- C64: 0801 - CFFF (50 kB)
- C16: 1001 - FBFF (59 kB)
- X16: 0801 - 9EFF (37¾ kB)
  - 9F00 - 9FFF I/O
  - A000 - BFFF banked RAM (512 kB or 2 MB)

C16: 13 kB leeway. X16: 8-9 kB missing

What if only interface word headers lived till end of load?

Idea: 2-staged heap

- local and global words
- re-using RAM for local word headers

Understanding | and CREATE - and CLEAR

```
: |  ?head @  ?exit  -1 ?head ! ;

: Create
  … ?head @ IF 1 ?head +! … heapmove $20 flag! … THEN … ;
```

- Hairy: REMOVE and ENDPOINTS called by CLEAR and FORGET
- Investigation outcome: New detailed code comments. No reuse.

Goal: Make it extensible at min extra cost

Breaking change

```
: Create  … ?head @ IF 1 ?head +! … heapmove THEN …
: |  ?head @  ?exit  -1 ?head ! ;
```

became

```
: Create  … ?headmove-xt @ IF … ?headmove-xt perform THEN …
: |            ['] heapmove1x ?headmove-xt ! ;
: heapmove1x       heapmove   ?headmove-xt off ;
```

New opportunity:

```
: |on          ['] heapmove ?headmove-xt ! ;
: |off                      ?headmove-xt off ;
```

tmpheap, x16tmpheap and notmpheap

```
User tmpheap[  User tmpheap>  User ]tmpheap
: mk-tmp-heap  ( size -- )  heap    dup ]tmpheap !
               tmpheap> !   hallot  heap tmpheap[ ! ;
: tmp-hallot   ( size -- addr )
               tmpheap> @ swap -  dup tmpheap> ! ;
: tmp-clear    remove-tmp-words  ]tmpheap @ tmpheap> ! ;
: ||           ['] tmp-heapmove1x ?headmove-xt ! ;
```

X16: `$a000 tmpheap[ !  $c000 dup ]tmpheap ! tmpheap> !`

Noop: `' | alias ||   ' noop alias tmp-clear`

Transient tmpheap and tmp6502asm

Tmpheap can live on heap:

```
here   $200 hallot  heap dp !
… ( tmpheap code ) …
dp !
```

Size of tmpheap on heap?

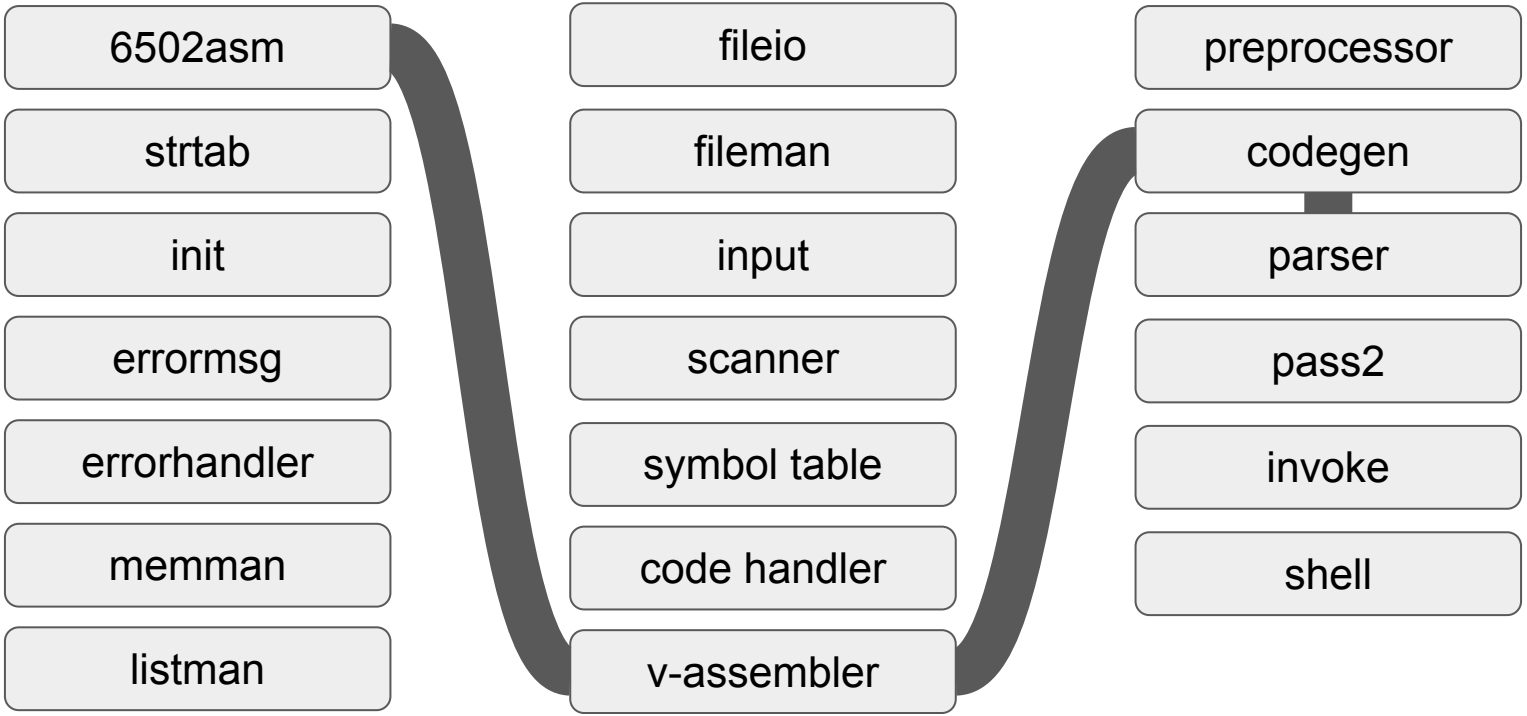
Assembler on tmpheap:

```
here   $800 tmp-hallot dp !
  include 6502asm.fth
dp !
```

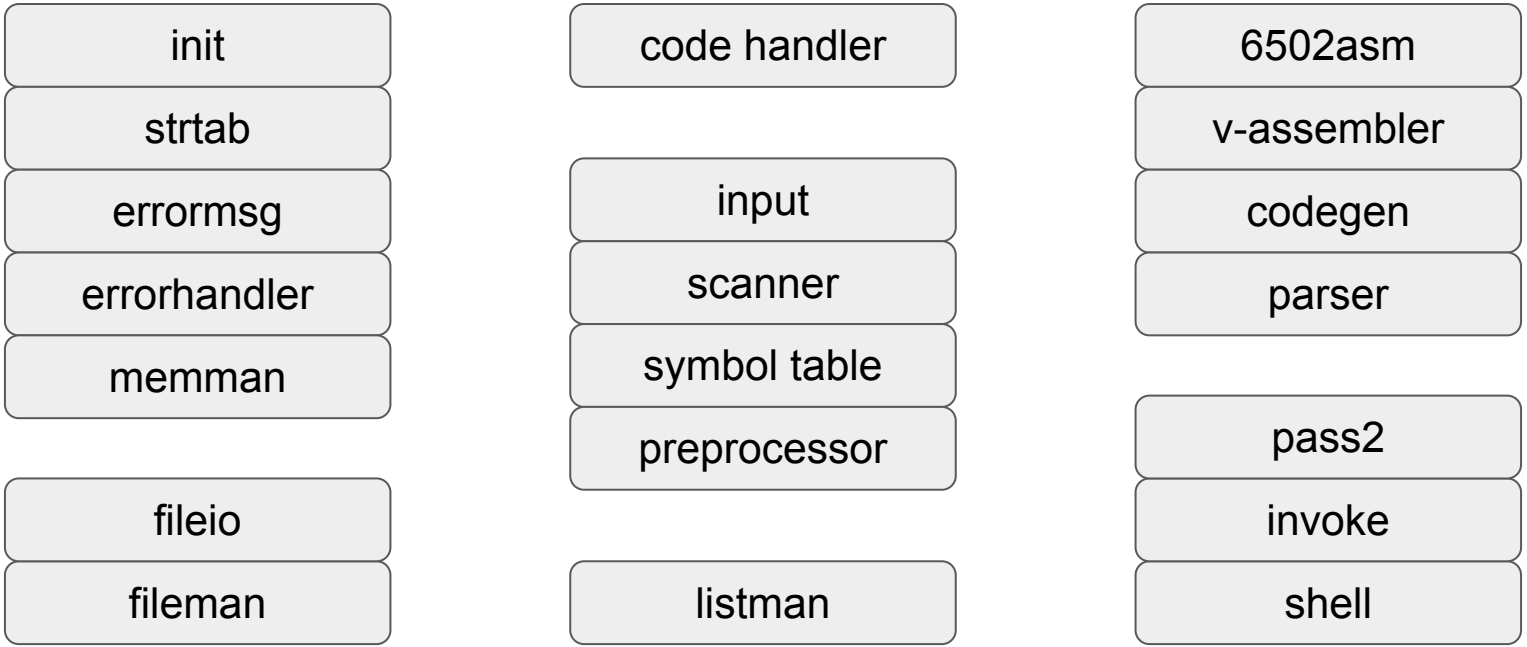
tmpheap size: tmp-clear after each module?

- 6502asm
- strtab
- init
- errormsg
- errorhandler
- memman
- listman
- fileio
- fileman
- input
- scanner
- symbol table
- code handler
- v-assembler
- preprocessor
- codegen
- parser
- pass2
- invoke
- shell

Problem: interface widths



Module groups - tmpheap 8 kB



Conclusions

- Feasibility of cc64 on C64 without headerless words doubtful
- Glad tmpheap reduced memory footprint enough for X16
- Enlightening to delve into a larger project’s dependency graph

Thank you for your attention!

Questions?